논문 2004-41SD-5-9

Mobile Multimedia 지원을 위한 Embedded Processor 구조 설계

(Design of Embedded Processor Architecture Applicable to Mobile Multimedia)

이 호 석*, 한 진 호*, 배 영 환*, 조 한 진*

(Ho Seok Lee, JinHo Han, Young Hwan Bae, and Han Jin Cho)

요 약

본 논문은 mobile platform에서 사용될 Multimedia 적용을 위한 embedded processor의 기본 구조 연구에 관한 내용으로 MPEG4 응용에 적합한 processor의 기본 구조 그리고 mobile platform에 적용될 수 있는 energy efficiency를 고려한 구조설계를 주 내용으로 하고 있다. multimedia 응용 embedded processor의 기본 구현 구조 요소인 processor data path architecture(pipeline, branch prediction, multiple issue superscalar, function unit number)의 기본 구조 설정과 cache hierarchy와 그 구성의 적합한 예상구조를 설정하기 위해 본 논문에서는 multimedia 응용 프로그램인 MPEG4를 processor simulator의 test bench로 사용하여 다양한 구조에 대한 simulation을 수행하였다. 그리고 mobile platform 적용에 적합한 구조인지에 대한 문제를 energy efficiency관점에서 고찰하여 적용 가능한 기본 processor 구조를 설정하였다. 그리고 본 논문에서 제안된 기본 구조 연구는 mobile platform에 바로 적용이 가능하며 더 나아가 특정 응용 프로그램에 최적의 성능을 발휘할 수 있는 자동화 설계기반환경에서의 configurable processor 설계에서 그 기본 processor 구조로 사용될 수 있다.

Abstract

This paper describes embedded processor architecture design which is applicable to multimedia in mobile platform. The main description is based on basic processor architecture and consideration about energy efficiency when used in mobile platform. To design processor data path architecture (pipeline, branch prediction, multiple issue superscalar, function unit number) which is optimal to multimedia application and cache hierarchy and its structure, we have run the simulation with variant architecture using MPEG4 test bench as multimedia application. We analyzed energy efficiency of architecture to check if it is applicable to mobile platform and decide basic processor architecture based on analysis result. The suggested basic processor architecture not only can be applied to mobile platform but also can be applied to basic processor architecture of configurable processor which is designed through automatic design environment.

Keywords: Processor, Multimedia, Embedded Processor, MPEG, Low Power

I. 서 론

Multimedia processor 설계 분야에서는 MPEG2와 같은 multimedia응용을 wired computing 환경에 구현하기에 적합한 구조 설계 관점에서 많은 선행 연구가 진행되었다[1].

고성능 PC에서 multimedia응용을 지원하기 위한 superscalar+SIMD CPU 구조[2][3][4][5]와 multimedia특성화된 VLIW+SIMD DSP 구조[6][7]등은 MPEG2응용을 일반적으로 사용되는 processor를 이용해 구현하여 실시간 동작을 목적으로 하고 있다. 현재 mobile platform 설계에 있어서 MPEG4 (QCIF(176X144), 128kbps)를 특성화 ASIC이 아닌 general processor 기반 구조로 구현 방식이 이동하고 있으며 이러한 구현의 일례로 ARM9+MOVE coprocessor[8], ARM9+ TMS320C55x coprocessor[9] 계열의 mobile 응용을 위한 구현 방식에서 확인할 수 있다. 이러한 구조는 고정된 응용만을 지원하는 ASIC구현에 비해 energy efficiency는 다소 떨어지지만 mobile환경에서 여러 응용 분야에 특화된 platform을 지원하며 특히 multimedia 응용에 있어서 다양한 변화를 지원할 수 있는 장점을 가지고 있다.

정회원, 한국전자통신연구원
(Electronics and Telecommunications Research Institute)
접수일자: 2003년2월28일, 수정완료일: 2004년4월20일

본 논문은 mobile platform에서 사용될 multimedia 적용을 위한 embedded processor의 기본 구조 연구에 관한 내용으로 multimedia 응용에 적합한 processor의 기본 구조 그리고 mobile platform에 적용될 수 있는 energy efficiency를 고려한 구조설계를 주 내용으로 하고 있다. processor의 구조 요소인 processor data path architecture(pipeline, branch prediction, multiple issue superscalar, function unit number)의 기본 구조 설정과 cache hierarchy와 그 구성의 적합한 예상구조를 설정하기 위해 본 논문에서는 multimedia 응용 프로그램인 MPEG4를 processor simulator 의 test bench로 사용하여 다양한 구조에 대한 simulation을 수행하였다. 그리고 mobile platform 적용에 적합한 구조 결정에 대한 문제를 energy efficiency 관점에서 고찰하여 가능한 processor 구조를 설정하였다. 그리고 본 논문에서 제안된 구조연구는 mobile platform에 바로 적용이 가능하며 더 나아가 특정 응용 프로그램에 최적의 성능을 발휘할 수 있는 자동화 설계기반환경에서의 configurable processor[10]설계에서 그 기본 processor구조로 사용될 수 있다.

본 논문의 구성은 Ⅱ장에서 processor 구조를 설정하기 위해 사용된 응용 프로그램 MPEG4 test bench의 특성과 사용된 simulator에 대한 간단한 소개를 Ⅲ장에서는 cache의 크기, set associative, block크기의 변화에 따른 hit rate, IPC 성능변화를 simulation한 결과를 정리하고 있으며 Ⅳ장에서는 branch predictor의 구현 종류와 그 크기 변화에 따른 성능변화와 multiple issue superscalar에서 RUU(Reservation Update Unit)의 크기에 따른 성능변화와 issue에 따른 성능변화에 대한 결과를 정리하고 Ⅴ장에서는 mobile platform하에서 적용 가능한 processor architecture를 가늠할 수 있는 energy efficiency에 대한 정의를 그리고 Ⅵ에서는 energy efficiency를 사용한 processor 구조들 간에 비교 그리고 mobile platform에 적합한 기본 processor 구조 제안으로 구성되어 있다.

Ⅱ. MPEG4 testbench와 simulation 환경

본 논문에서 사용된 simulator는 simplescalar[11]로 MIPS-IV ISA(Instruction Set Architecture)를 바탕으로 하며 single/double precision floating point square roots 연산과 함께 load와 store시는 두 가지 address mode(index(register+register), auto increment/decrement)를 부가 지원하고 register/immediate/jump의 3가지 64-bit의 instruction format을 가진다. 그리고 integer register의 경우 32-bit를 갖는 32개의 general register와 program counter(PC), multiplier용 HI/LO result register, floating point condition code(FCC)로 구성되어 있으며 floating point register는 32-bit(single precision)크기의 32개 또는 64-bit (double precision)크기의 16개로 조합 가능한 register로 구성되어 있다. 그리고 test bench로 사용된 MPEG4 프로그램은 ETRI에서 독자적으로 개발한 simple profile level 2 (QCIF(176X144), 128kbps)의 MPEG4 ASIC[12]의 검증과정에서 사용된 reference C 코드로 사용된 MOVA 소프트웨어를 사용하였으며 그 구성은 대략적으로 그림 1 과 같다.

MOVA의 Compile된 파일의 특성은 표 1에 간략히 정리되어 있다. 결과 중 몇 가지 참고할 내용은 우선 UltraSparc용 VIS(Visual Instruction Set)의 instruction set 및 compiler가 simplescalar에서 지원하는 MIPS-IV에 비해 코드 사이즈에서 2.5배 우수하다는 것을 알 수 있으며 본 논문에서 사용된 MOVA의 encoding압축률이 78배가 됨을 알 수 있다.

그리고 real stream환경이 아닌 상황에서 입력 신호 파일은 compile과정에서 byte단위의 영상 신호를 integer format(4byte)으로 저장되어 profiler의 memory내에서 data 크기가 4배 증가하게 되었다.

실제 processor구현 후 real 환경 동작에서는 일반적으로 2 frame을 이용하므로 10 frame을 사용하여 compile한 결과에 비해 1/5의 data memory가 사용됨을 확인하여야 한다.

Ⅲ. Cache hierarchy

Processor와 main memory사이에 위치하는 저장 공간인 cache의 설계 시 고려하여야 하는 중요한 점은

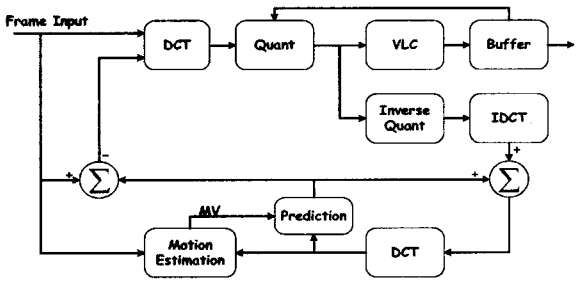


그림 1. MPEG4 인코딩 블록도
Fig. 1. MPEG4 Encoding Diagram.

표 1. MPEG4 인코딩 테스트벤치 특성
Table 1. MPEG4 encoding test bench characteristics.

Characteristics Items	Size
Encoding input file	38016 Byte/frame
Encoded output file	482 Byte/frame
Compiled MPEG4 text size when use UltraSPARC gcc compiler(VIS)	293928 Byte
Compiled MPEG4 text size when use simplescalar compiler(MIP-IV)	734884 Byte
MPEG4 program text size in simplescalar profiler	362736 Byte
MPEG4 program data size in simplescalar profiler when run 10 frames	1609556 Byte

표 2. data path 설정 계수
Table 2. data path parameter.

data path 구조		설정 계수
Branch direction prediction hit rate		0.85
Instruction decode, issue, commit		8
RUU		256
Functional Unit	integer ALU, multiplier,divider	8
	floating point ALU, multiplier,divider	8

cache의 크기와 miss rate를 최소화할 수 있는 구조로 설계하는 것이다. 일반적으로 cache의 크기가 클수록 miss rate는 줄어들지만 first chunk access latency가 커져 전체성능 개선은 줄어든다. 그러므로 cache는 first chunk access latency를 최소화하면서 miss rate를 개선할 수 있도록 설계되어야 한다. 특히 본 processor 설계에서는 특정 응용(ex.. MPEG4)에 특화된 접근을 하므로 cache의 구현 구조인 block size, set의 entry개수, set associative, replacement strategy의 설정에 따라 최적의 성능을 발휘하도록 할 수 있다.

이러한 근거로 cache simulation은 우선 set associa-tive에 따른 성능 변화를 살펴보고 이를 바탕으로 cache의 block 크기와 set entry 수를 변화시키는(cache 크기 변화) simulation을 수행하였다. 또한 MPEG4 프로그램에서 발생한 cache miss의 발생요인에 대한 분석을 다루고 있다. 본 장에서 수행된 simulation에서는 cache를 대상으로 한 성능 변화를 좀 더 정확히 측정하기위해 processor data path에 의한 성능의 영향을 없애야 하였고, data path의 모든 설정계수의 크기를 충분하도록 설정하였다. 그 설정 계수는 표 2 와 같다.

Cache의 hit latency 및 miss penalty는 cache 크기 및 block 크기에 의해 결정되며 설정된 latency는 hit

표 3. cache 모델링 가정
Table 3. Cache modeling assumption.

Cache Feature	Assumption
Clock_Cycle_Time	2ns/1clock(500Mhz)
Cache_Bandwidth(Processor-L1)	8Byte(64bits bus)
Bus_Bandwidth (L1-Main Memory)	8Byte (64bits bus)
Row_Counter of cache memory(SRAM)	Cache_Size/Cache_Bandwidth
Gate_Transition_Delay	0.2ns(0.25um tech)
Transition delay of row line of cache memory(SRAM)	Tc*(Row_decoder_transition), Tc=1

rate의 계산과는 무관하지만 전체 성능분석 변수인CPI (Clock per instruction) 또는 IPC(Instruction per clock)에 영향을 주는 계수이다. 수행된 simulation에서 설정한 L1 cache의 hit latency설정은 표 3에 기술한 내용을 원칙으로 한다.

표 3의 가정에서 설정된 값에 따라 L1 cache hit latency는

$$\text{Cache Hit Latency} = \left\lceil \frac{Tc \times Gate\_Transition\_Delay \times \log_2^{Row\_Count}}{Clock\_Cycle\_Time} \right\rceil$$

와 같이 근사화 될 수 있으며 앞서 정의된 전제에 따라 계산하면 32kB이하에서는 L1 cache hit latency가 1로 그 이상에서는 2로 설정된다. 그리고 main memory는 off-chip으로 bus bandwidth를 8Byte로 제한하였으며 이에 따라 L1 cache miss penalty는 다음과 같다.

$$\text{L1 Cache Miss Penalty} = First\_Chunk\_Delay + \frac{Cache\_Block\_Size \times Inter\_Chunk\_Delay}{Bus\_Bandwidth}$$

그리고 L2 cache를 사용하지 않을 경우 first chunk delay를 18로 그리고 inter chunk delay를 2로 설정하였다. 앞서 설정한 model은 transition coefficient(Tc)를 이용한 gate level의 대략적인 modeling방법을 사용한 것으로 보다 정밀한 model을 위해서는 transistor level의 circuit modeling을 사용할 수 있다[13].

그림 2는 L1 cache의 구조 중 set associative와 block크기에 따라 변화시키면서 L1 cache에서 발생하는 miss rate와 IPC(1/CPI)를 정리한 그림이다. 전체 entry의 수는 128로 고정되어 있는 상태에서 set associative를 변화시키면서 simulation한 결과 set associative가 2(set 당 entry수는 64)일 경우가 direct mapped 방식에 비해 cache miss rate가 감소하고 IPC가 증가한다. 그 이상에서는 개선되지 않는다. 이러한 결과로 볼 때 사용된 test bench의 instruction 및 data access시 conflict에 의한 miss 발생이 적음을 확인할 수 있다. 그

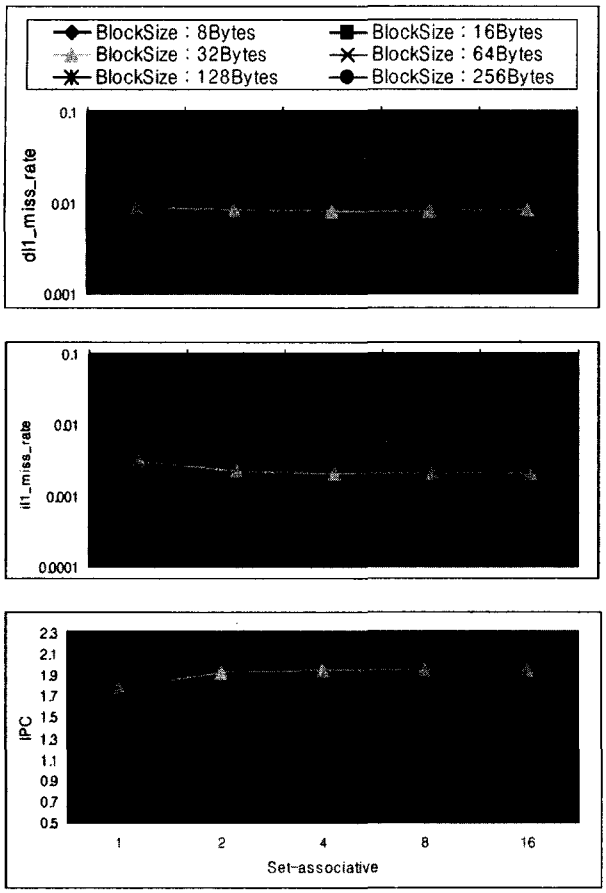


그림 2. L1 cache의 set associative 와 block 크기 변화에 따른 cache miss rate와 IPC 값

Fig. 2. Cache miss rate and IPC according to set associative and block size change of L1 cache having the following Spec.
Total entry number of cache: 128(entry/set x set associative), Cache size: 1kB(when block size is 8 Byte), 2kB(16Byte), 4kB(32Byte), 8kB(64Byte), 16kB(128Byte), 32kB(256Byte).

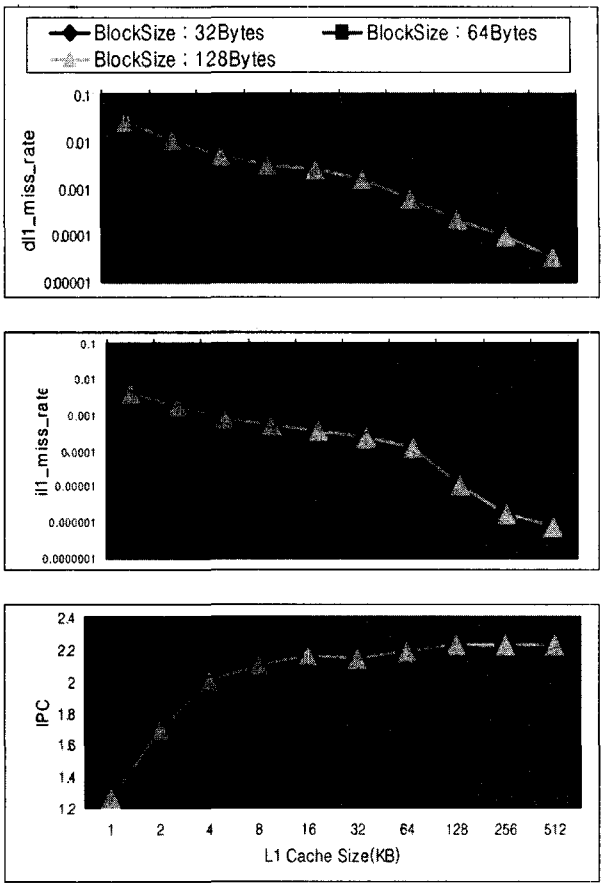


그림 3. L1 cache의 크기에 따른 miss rate와 IPC

Fig. 3. L1 cache miss rate and IPC according to cache size change.

리고 block의 크기와 cache 크기를 함께 증가시키는 상황에서 block의 크기가 128Byte(16kB cache size)이상은 개선비가 줄어들지만 전체적으로 cache의 크기를 2배씩 증가함에 따라 miss rate도 2배 이상으로 감소한다는 것을 확인 할 수 있었다.

그리고, 그림 2의 세 번 째 그래프는 set associative와 block 크기를 변화에 따른 L1 data/instruction cache의 miss rate, hit latency 그리고 앞에서 설정한 processor datapath의 영향으로 다음과 같이 변화하는 IPC (Instruction per clock)를 정리한 것이다.

$$\frac{1}{IPC} = CPI = CPI_{Execution} + \frac{\text{Memory access}}{\text{Instruction}} \times \text{Miss rate} \times \text{Miss penalty}$$

miss penalty는 앞서 소개한 수식에 따라 18+(cache_block_size/8)/2 가 된다. block 크기(cache_block_size)가 8, 16, 32, 64, 128, 256 byte에 따라 miss penalty는 18, 20, 24, 32, 48, 80 cycle이 된다. 이러한 miss penalty의 영향은 IPC에 직접적으로 영향을 주며 그 영향은 그림 2의 IPC 그래프에서 확인할 수 있다.

Cache의 구조 변화가 전체성능에 주는 영향을 정리하겠다. block의 크기와 cache의 크기를 함께 증가시키면 spatial locality가 향상되고 miss rate도 줄어들어 전체성능이 개선될 것으로 예상하였으나 cache miss시 update해야 할 data가 많아져 miss penalty가 증가된다. 특히 64byte block이상에서는 cache가 커지더라도 성능 향상의 개선 정도가 아주 작다.

MPEG4 프로그램을 수행시 cache의 전체 entry개수는 128로 고정하면 HW 복잡성과 성능의 trade off에서 2 set associative가 적당하고 miss rate와 IPC의 trade off에서 32, 64, 128 byte block이 적합하다는 것을 확인하였다.

그림 3에서는 2 set associative로 고정하고 32, 64, 128 byte block을 대상으로 entry 수와 cache 크기를 함께 증가시키면서 instruction cache와 data cache의

성능을 좀 더 정밀하게 측정하였다.

cache 크기가 같으면 32 byte block일 때의 entry 수는 64 byte block일 때 보다 2배가 되고 128 byte block을 사용할 때보다 4배가 된다. 그림 3에서 miss rate의 전체적인 동향을 보면 cache 크기가 동일할 때 큰 block을 사용할수록 cache miss가 줄어드는 것을 확인할 수 있다.

Data cache의 경우 동일한 cache 크기일 때 사용할 수 있는 entry 수가 제한적인 4kB 크기 이하에서는 (block 크기가 32 byte일 때 64 entry, 64 byte일 때 32 entry, 128 byte일 때 16 entry이다.) entry의 수에 따라 cache miss는 불규칙적이지만 전체적으로 보면 instruction cache와 같이 cache 크기가 동일할 때 큰 block을 사용할수록 cache miss가 줄어드는 것을 확인할 수 있다. data cache의 entry수를 최소로 한다는 의미는 많은 data를 필요로 하는 코드는 여러 block에 나누어져 있는 data를 읽을 수 있다. MPEG4의 주요하위 기능에는 motion estimation 또는 motion compensation에서 다른 frame에 위치하는 macro block들을 서로 비교하는 동작이 이에 해당한다.

miss rate와 hit latency 그리고 processor data path에 따라 얻은 IPC를 볼 때 block크기가 64byte일 때 우수한 성능을 보이고 있으며 cache 크기가 클수록 IPC가 커지지만 hit latency가 2가 되는 32kB이상의 cache를 사용하였을 때는 성능향상이 미미하고 32kB에서는 오히려 성능이 저하된다. 그리고 HW 복잡성과 성능의 trade off로 16kB의 cache를 사용하는 것이(hit latency가 1인 cache를 사용하였을 때) 가장 적합하다. 위와 같이 cache를 키울수록 증가하는 hit latency를 줄이기 위해 line buffer 또는 pipelined cache를 사용하는 경우가 있다. 그렇지만 MPEG4와 같은 작은 프로그램일 때는 이러한 기법에 의한 성능향상이 미미하다[14].

지금까지 cache simulation에서는 set associative에 따른 성능 변화를 살펴보고 이를 바탕으로 block 크기와 set의 entry를 변화시키는(cache크기변화)simulation을 하였다. MPEG4 응용 프로그램을 이용할 경우 2 set associative가 HW 복잡성과 성능의 trade off에서 유리하다는 것과 32, 64, 128 byte크기의 block이 miss rate와 IPC에서 적합하고 이중 64 byte block이 전체 성능면에서 가장 적합하다. 그리고 최소 first chunk access latency를 유지하는 최대 cache 크기(ex. 16kB)를 사용시 최상의 성능을 가진다는 결론을 얻었다.

IV. Branch prediction과 multiple issue

Superscalar가 일반화됨에 따라 branch prediction 기술은 processor의 성능을 많은 영향을 미친다. Branch prediction은 branch의 방향(taken 또는 not-taken)예측과 taken branch인 경우 최소의 delay로 branch target instruction를 구하는 두 가지 기술로 나눌 수 있다. Target instruction을 얻는 방법으로 target instruction을 미리 저장해 두는 특별한 cache인 BTB(Branch target buffer)를 사용하는 경우이다. 가장 잘 알려진 prediction방법으로는 최근에 사용된 branch의 direction을 바탕으로 예측을 수행하는 기본적인 bimodal branch prediction과 각 branch의 history를 독립적으로 이용하고 branch의 반복적인 pattern을 갖는 특징을 이용하는 local branch prediction(history가 독립적이므로)과 최근 branch들의 상호 history를 사용하는 global branch prediction을 들 수 있다[15][16].

구현 시 적용될 수 있는 다양한 branch predictor에 따른 hit rate와 IPC를 다루고자 한다. simulation에서 memory access 및 multiple issue에 관계된 계수는 이상적으로 설정하여 branch prediction에 미치는 영향을 최소화 하였다.

그림 4에서는 cache simulation에서 사용된 MOVA를 사용하여 앞서 언급한 bimodal branch prediction, global branch prediction, local branch prediction 그리고combined branch prediction에 대해 동일한 구현 크기를 가질 때의 branch prediction hit rate와 IPC를 구한 것이다. 첫 번째 그래프에서 가로축의 구현크기(table size)는 각 구현방식에 따라 사용되는 buffer(BHT, PHT)를 정량화하여 계산한 것으로 각 구현방식에 따라 구현크기 계산은 표 4에 정리되어 있다.

MOVA 사용시 동일한 구현 크기일 때 GAg방식이 전체적으로 가장 열악한 hit rate와 IPC성능을 보이며 이때 table크기를 늘려도 성능향상에 한계를 보인다. 이러한 성능제한은 여러 branch instruction이 하나의 branch history table(BHT)와 pattern history table(PHT)을 함께 쓰는 것에 의한 구현 구조적인 문제에 의한 결과이다. 구현 크기가 작은 경우에 bimodal, gshare, PAg방식이 구현 HW를 가장 많이 사용하는 PAp방식에 비해 우수한 성능을 보이며 구현 크기가 커질수록 이러한 경향은 반전되어 PAp 방식이 앞서 언급한 방식에 비해 우수한 성능을 보인다. 그리고 combined 방식의 일례인 bimodal/gshare방식은 구현

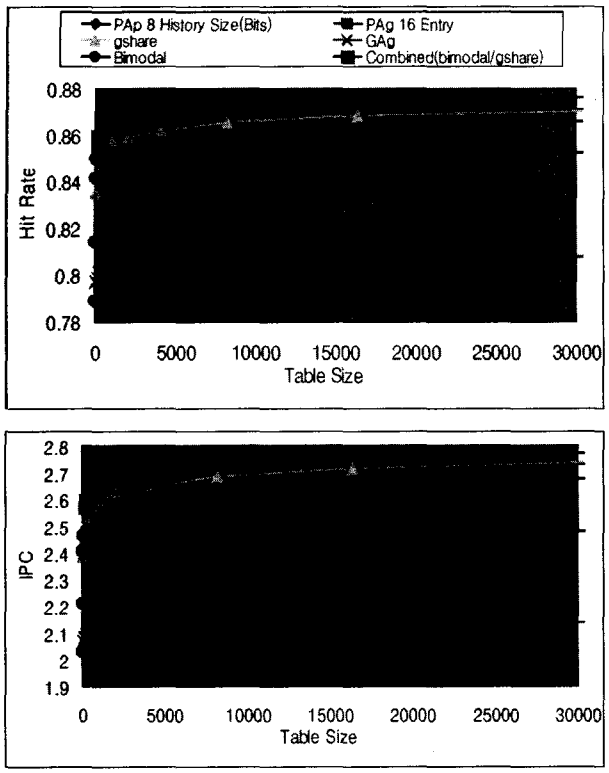


그림 4. 동일한 크기를 가질 때 branch prediction방식에 따른 성능비교

Fig. 4. Performance comparison depend on branch prediction scheme at same area.

표 4. branch predictor의 구현 크기

Table 4. estimated cost of branch predictors.

Scheme Name	BHT Register Length	Number of PHT	Simplified Hardware Cost
GAg(k)	k	1	$k + 2^k \times 2$
PAg(k)	k	b	$b \times k + 2^k \times 2$
PAp(k)	k	b	$b \times k + b \times 2^k \times 2$
gshare	k	1	$k + 2^k \times 2$

크기에 무관하게 전체적으로 다른 구현 방식에 비해 우월한 성능을 보이고 있다. 결과적으로 각 branch가 특정 방향으로 편향되어 있을 경우 유리한 bimodal bran-ch prediction과 branch들이 단순한 반복적인 pattern을 가질 때 유리한 local branch prediction 그리고 연속적으로 수행되는 branch의 direction이 서로 상관성이 높을 경우에 유리한 global branch prediction의 조합으로 구현된 combined branch prediction방식이 어떤 pre-dictor의 결과를 선택할지를 결정하는 부가적인 predic-tion selector구현에도 불구하고 구현 면적 대비 성능의 효율이 가장 우수하다.

다수의 function unit와 instruction queue와 같은 unit를 사용하여 한 clock당 동시에 여러 개의 instruc

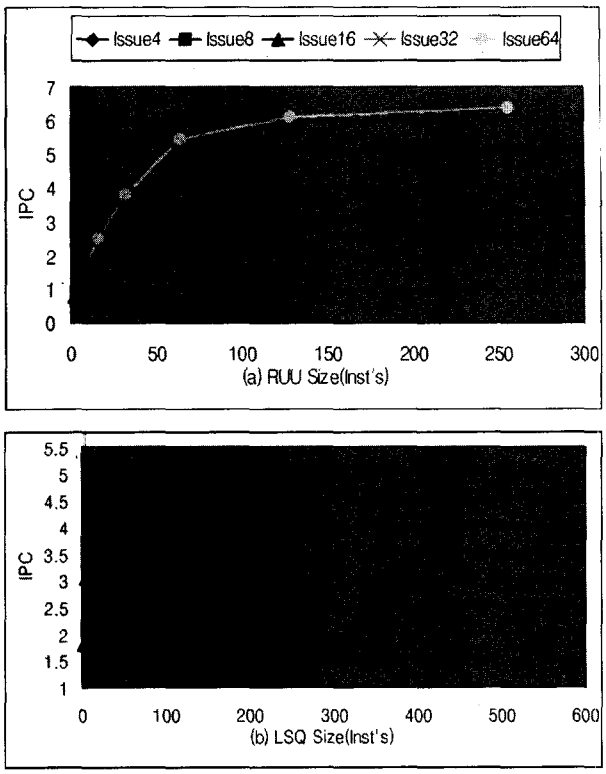


그림 5. RUU/LSQ 크기변화에 따른IPC 성능

Fig. 5. IPC performance according to RUU/LSQ size change.

-tion을 처리하고자 하는 multiple issue superscalar 동작은 H/W기반에서 instruction level parallelism을 이용한 것으로 MOVA와 같은 multimedia 응용 프로그램의 경우 동시에 수행 가능한 instruction이 풍부해 multiple issue 동작의 경우 다른 응용에 비해 우수한 성능향상을 기대할 수 있다. 본 논문에서는 superscalar구현 방식에 있어서 RUU(Register Update Unit)방식을 이용한다[17]. 앞서 branch prediction simulation에서는 multi-ple issue와 관계된 하위 구조의 구성을 최대화하여 multiple issue에 따른 simulation에 미치는 영향을 최소화하였다. 이와는 반대로 multiple issue simulation에서는 branch prediction을 perfect로 설정하여 multiple iss-ue simulation에 미치는 영향을 최소화하였다.

그림 5의 (a)는 issue bandwidth를 4, 8, 16, 32, 64 instruction으로 설정될 때 RUU크기에 따른 성능 변화를 보여 주고 있으며 (b)는 issue bandwidth를 16 instruction으로 설정하고 LSQ크기에 따른 IPC성능변화를 나타내고 있다. (a) 결과에서 32, 64 issue를 수행하더라도 issue가 16일 때에 비해 더 이상의 성능 개선이 없음을 확인할 수 있으며 이러한 특성은 MPEG4 test bench의 instruction parallelism의 한계를 보여 준

다. 그리고 특정 issue에 대해 RUU size를 한계이상으로 증가시키더라도 더 이상의 성능개선은 보이지 않음을 확인할 수 있다. 그림 5에서 보이고 있는 결과는 perfect branch prediction의 수행을 가정하고 있어 성능 향상 saturation point가 높게 설정된 결과를 얻었으나 실제 구현에서는 not-perfect branch prediction과 memory miss로 인해 issue에 따른 그리고 queue크기에 의한 성능 향상 saturation point는 더 낮은 지점에서 형성된다. 특정 구조의 superscalar방식 processor를 구현할 경우 branch predictor 및 multiple issue의 구현에 소모되는 HW 및 이에 따른 성능향상을 IPC값으로 구할 수 있으며 이러한 결과는 설계 설정에 중요한 평가 수치로 사용된다.

V. Energy Efficiency

본 논문에서 중점적으로 기술하는 mobile용 embedded processor의 설계에서는 앞장에서 수행된 구현구조에 따른 성능 변화에 못지않게 중요한 변수로 energy efficiency를 들 수 있다. 앞서 수행한 simulation결과로 전체 성능을 크게 향상시키는 구조일지라도 energy efficiency가 좋지 않은 구조는 실제 구현에 앞서 재고의 여지가 있다. Energy efficiency를 고려하기 위해서는 energy efficiency를 측정하기 위한 measure unit 정의가 선행되어야 하며 power, energy, energy-delay등의 unit가 고려될 수 있다. 기존의 타기종간 성능비교로 사용되는 단위인 power는 processor clock frequency에 비례하고, processor에서 소모하는 energy는 processor의 물리적 특성에 의해 좌우되므로 energy(jule / instruction)는 다양한 구현 구조간의 energy efficiency를 비교하기에는 적합하지 않으며 energy-delay product를 이용한 단위를 사용할 경우 특정 응용 프로그램이 소비하는 energy와 프로그램이 수행되는 시간을 포함하고 있어 구현구조의 효율을 다루기에 적합하다. 이러한 이유로 본 논문에서는 processor구조의 적합성을 energy-delay product를 사용하여 평가하였다. 그리고 본 논문에서는 공정에 의한 성능개선을 제외하고 구조적인 관점에서 설계개선을 고려할 수 있도록 energy-delay product를 상호 비율로 설정하였고 energy-delay product는 다음과 같이 정리될 수 있다.

$$\text{Energy - Delay} = \text{Energy} \times \text{CPU}_{\text{Time}}$$

$$\text{Energy} = \sum_{i}^{N} \left(\frac{1}{2} \times \alpha_i \times C_i V^2\right)$$

$$\text{CPU}_{\text{Time}} = \text{IC} \times \text{CPI} \times \text{Time}_{\text{Clock}} = \frac{\text{IC} \times \text{Time}_{\text{Clock}}}{\text{IPC}}$$

$$\text{IC} = \text{Compiled Instruction Number}$$

$$\frac{1}{\text{IPC}} = \text{CPI} = \text{CPI}_{\text{Execution}} + \frac{\text{Memory access}}{\text{Instruction}} \times \text{Miss rate} \times \text{Miss penalty}$$

앞서 계산된energy-dealy product에서 Compiler의 성능에 의해 좌우되는 IC는 고정된 값으로 설정될 수 있고 unpipelined구조가 아닌 simple pipeline/superscalar의 구조를 비교할 경우 $\text{Time}_{\text{clock}}$ 역시 공정에 의해 좌우되는 scaling factor의 변수로 간주하여 고정시킬 수 있다. 이러한 가정을 바탕으로 energy -delay efficiency는 다음과 같이 간략하게 기술된다.

$$\text{Energy} - \text{Delay Efficiency} = \frac{\text{Energy}}{\text{IPC}}$$

기존의 연구에서 superscalar방식의 동작 시 각 구성요소에서 소모되는 전력에 관한 다수의 연구가 선행되었으며 power model을 바탕으로 정밀한 energy 소비에 대한 비율 구할 수 있다[18].

표 5는 superscalar방식으로 구현된 processor에서 소모되는 energy를 각 하위 구성단위 별로 정리한 표로 simulation에 사용된 중요 하위 구성단위의 구조 설정은 다음과 같다.

Fetch/issue width: 4 instructions
Instruction Queue size: 128 entries
Functional units: 4 intALU, 4 fp ALU, 1 int mul/div, 1 fp mul/div
Branch predictor: Combined (1k entry gshare/8bit global history/2k entry bimodal/1k entry selector)
I-cache L1: 128kB, direct mapped, 32byte block, 1 cycle hit, 3 cycle miss penalty
D-cache L1: 128kB, 4-way set associative, 32byte block, 1cycle hit, 3 cycle miss penalty
I/D-cache L2: 1MB, 4-way set associative, 64byte block, 3cycle hit, 16 cycle first chunk, 2 cycle interchunk

표 5에서 볼 때 많은 전력이 instruction queue와 이에 부가된 회로에서 소모됨을 알 수 있고 이는 전체 소모전력의 대략 25%에 해당한다. 그리고 reorder buffer에서 역시 instruction queue와 비슷한 정도의 전력을 소모한다. 이러한 superscalar 동작에 의한 전력소모는 모든 instruction이 issue rate에 비례하여 매 cycle마다 모든 buffer를 access하기 때문이다. 그리고 표 5는 4

표 5. superscalar에서의 energy소모 비율
Table 5. Energy consumption ratio in superscalar.

Sub-Component	Average Power consumption(%)
Inst. decode	2.8
BTB	1.1
TLB	0.5
L1 I/D-cache	10.4
L2 Cache	11.1
Rename table	13.8
Inst. Queue	26.5
ROB	27.1
Function Unit	2.0
I/O logic	2.0
Other parts	2.7

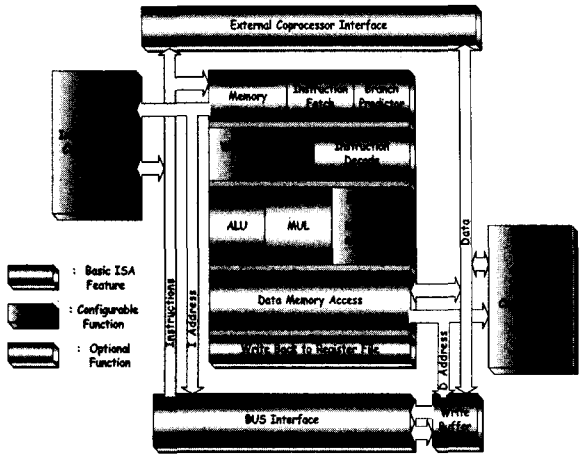


그림 6. configurable processor 구조
Fig. 6. configurable processor architecture.

표 6. 제안된 Embedded Processor 구조
Table 6. Proposed Embedded Processor Architecture.

Architecture Feature	Simple Pipeline	SuperScalar Pipeline	Suggested Architecture
Basic Feature	5 stage	6 stage	5 stage
Cache	Cache Size : 16KByte, Block Size : 64Byte, Set Associative : 2-set	Cache Size : 16KByte, Block Size : 64Byte, Set Associative : 2-set	Cache Size : 16KByte, Block Size : 64Byte, Set Associative : 2-set
Branch Prediction	Static Hit rate : 0.21 IPC : 0.54	Dynamic Combined (Bimodal/gshare) Hit rate : 0.87 IPC : 1.68	Dynamic Combined (Bimodal/gshare) Hit rate : 0.90 IPC : 0.79
Out-of-order Execution	1 Issue	4 Issue	1 Issue
Energy Consumption Ratio	100 Inst.decode : 14 L1$, TLB : 52 Function Unit. et al: 34	1735 Inst.decode : 56 L1$, TLB : 208 Function Unit. et al : 105 RUU: 1344 BTB : 22	122 Inst.decode : 14 L1 $, TLB : 52 Function Unit. et al : 34 BTB : 22
Energy Delay Product efficiency	1	5.58	0.83

issue를 가정한 결과로 superscalar형태를 가지지 않는 processor가 1 issue임을 가정할 때 superscalar 방식으로 구현된 모든 block의 동작은 기본적으로 수배에 달하며 이를 고려하면 superscalar를 구현하기 위해 사용된 하위 구성단위에서 소모되는 energy 증가는 1 issue 기본 pipeline구조에 비해 매우 크다. 그리고 관심의 대상 중 하나인 branch predictor에서 소비되는 energy는 구현구조와 전체 instruction에서 차지하는 branch instruction의 비율에 의해 결정되며 MOVA의 경우 전체 instruction에서 branch instruction이 17%내외의 비율을 가져 branch predictor의 하위 구성단위인 BTB(branch target buffer)에서 소모되는 energy는 전체 소모 energy중 1.1%의 미미한 energy만을 소비한다.

VI. Embedded Processor Architecture

III, IV장에서는 MOVA를 이용한 설계구조에 따른 성능 및 그 최적의 구현 방식을 부분적으로 산출하였다. 그리고 V장에서는 energy efficiency에 대한 정의를 설정하고 multiple issue superscalar 방식으로 구현된 processor의 각 하위 구성단위에서 소모되는 energy의 비율에 대해 기술하였다. 이러한 결과를 바탕으로 본 장에서는 mobile multimedia 지원을 위한 energy efficiency가 우수한 구조를 도출하고 있다. V장에서 제시한 processor 하위 구성단위에서 소비되는 energy의 비율과 III, IV장에서 구한 구조변화에 따른 IPC 값을 이용하여 processor의 기본 구조 구성에 따른 energy-delay efficiency를 구할 수 있다. 이를 이용하여

기존의 1 issue simple pipeline 구조와 4 issue superscalar 구조 그리고 branch predictor를 사용하는 1 issue simple pipeline 구조를 비교하였다. 표 6에서 static branch prediction을 사용하는 simple pipeline 구조와 dynamic branch prediction과 4 issue를 수행하는 superscalar 구조 그리고 dynamic branch prediction을 수행하는 1 issue pipeline 구조에 대한 energy-delay efficiency를 정리하고 있다. Simple pipeline에서는 taken 또는 not-taken의 설정에 따라 동작하는 static branch prediction을 사용하여 branch의 동작이 편향된 방향으로 설계된 경우 branch prediction의 hit rate가 0.21로 나타났으며 이때 dynamic branch prediction을 사용하면 같은 test bench에 대한 hit rate가 0.9에 가까이 도달한다. 이때 구한 IPC는 표 6에 설정된 cache의 구조와 not perfect prediction에 의해 제한되어 IV에서 제시된 simulation결과에 비해 다소 작은 값을 보이고 있다. 그리고 energy 소모 값은 simple pipeline을 기준으로(100) 설정하여 앞서 제시한 표 5에 따라 비율로 환산한 값을 각각 나타내고 있으며 이에 따라 계산된 4 issue superscalar의 energy소모 값은 simple pipeline에 비해17배에 달하고 하위 구성단위에 대한 각각의 환산 값은 표 6에 정리하였다. 그러나 dynamic branch pre-diction을 사용하는 1 issue pipeline구조에서는 22%의 energy 소모 증가만을 보인다. 이러한 결과를 바탕으로 앞서 도출한 간략한 energy-delay efficiency를 사용하여 그 효율을 구하면 static branch prediction을 사용한 simple pipeline 구조를 1로 기준으로 할 경우 4 issue superscalar 구조는 5.6배 열악한 효율을 그리고 dynamic branch prediction을 수행하는 1 issue pipeline 방식은 17%로 개선된 효율을 보인다. 이러한 결과로 전력소모가 중요한 문제가 되는 embedded processor가 multimedia를 지원하기 위한 구조로는 dynamic branch prediction 구조를 사용하는 1 issue pipeline 구조가 가장 효과적임을 확인할 수 있다.

VII. 결 론

본 논문에서는 multimedia 응용 프로그램인 MPEG4 test bench를 사용하여 cache hierarchy와 이에 따른 다수의 구현구조 simulation과 processor data path에서의 다수의 branch prediction 방식과 multiple issue superscalar방식에 대한 simulation을 수행하였다. 그리고 이러한 결과를 바탕으로 저전력소모가 절대적으로 필요한 mobile platform하에서 multimedia지원을 위해 앞선 결과로 도출된 구조가 적합한지에 대한 문제를 전력효율의 관점에서 다루고 그 결과를 바탕으로 향후 mobile용 embedded processor의 설계 시 그 기본이 되는 구조를 도출하였다. 도출된 기본 processor구조는 mobile용 embedded processor설계의 기본 구조로 사용될 수 있으며 그림 6에서와 같이 configurable proce-ssor의 구현에 있어서 기본구조로 응용될 수 있다. 이러한 기본 구조는 본 논문에서 다루는 범위는 아니지만 향후 설계 및 확장예정인 instruction set과 선택적 설계가 가능한 coprocessor 확장 설계의 기본 구조로 도입할 예정이다.

참 고 문 헌

[1] I. Kuroda and T. Nishitani, "Multimedia Processors", Proceeding of the IEEE, Vol.86, No.6, pp.1203-1221, June 1998.

[2] A. Peleg et al, "Intel MMX for Multimedia PCs", Communication of The ACM, Vol.40, No.1, pp.25-38, January 1997.

[3] "MIPS Extension for Digital Media with 3D", MIPS Technologies, Inc, March 1997.

[4] M. Tremblay et al, "VIS Speeds New Media Processing", IEEE Micro, Vol.16, No.4, pp.10-20, August 1996.

[5] R. B. Lee, "Sub-word Parallelism with MAX-2", IEEE Micro, Vol.16, No.4, pp.51-59, August 1996.

[6] P. Foley, "The Mpact Media Processor Redefined the Multimedia PC", Proceedings of COMPCON 1996, pp.311-318.

[7] C. Hansen, "Architecture of a Broadband Media Processor", Proceedings of COMPCON 1996, pp.334-340.

[8] "PrimeXsys Wireless Platform", http://www.arm.com

[9] S. Ramamurthi, "Multimedia Technologies on Terminals Based on the OMAP Platform", TI White Paper, June 2002.

[10] R. E. Gonzalez, "Xtensa: A Configurable and Extensible Processor", IEEE Micro, Vol1.20, No.2, pp.60-70, April 2000.

[11] T. Austin et al, "simplescalar: An Infrastructure for Computer System Modeling", IEEE Computer,

Vol.35, No.2, pp.59-67, February 2002.

[12] S. Park and et al, "A MPEG-4 Video Codec Chip with Low Power Scheme for Mobile Application", International Conference On Circuits/Systems, Computers and Communications 2002, pp.1288-1291.

[13] S. J. E. Wilton and N. P. Jouppi, "CACTI: An Enhanced Cache Access and Cycle Time Model", IEEE JSSC, Vol.32, No.5, pp.677-688, May 1996.

[14] K. M. Wilson and K. Olukotun, "Designing High Bandwidth On-Chip Caches", Proceedings of the 24thInternational Symposium on Computer Architecture, June 1997.

[15] S. McFarling, "Combining Branch Predictors", Tech. Note TN-36, DEC WRL, June 1993.

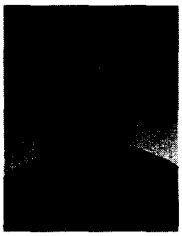
[16] T. Yeh and Y. N. Patt, "A Comparison of Dynamic Branch Predictors that use Two Levels of Branch History", Proceedings of the 20th International Symposium on Computer Architecture, pp.257-266, May 1993.

[17] G. S. Sohi and S. Vajapeyam, "Instruction issue logic for high-performance interruptible pipelined processors", Proceedings of the 14th Annual Symposium on Computer Architecture, pp.27-34, June 1987.

[18] D. Folegnani and A. Gonzalez, "Energy-Effective Issue Logic", Proceedings of the 28th Int. Symposium on Computer Architecture, June 2001.

[19] R. Gonzalez and M. Horowitz, "Energy Dissipation In General Purpose Microprocessors", IEEE JSSC, Vol.31, No.9, pp.1277-1284, September 1996

저 자 소 개



이 호 석(정회원)

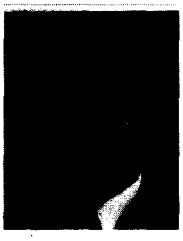
1996년 경북대학교 전자공학과 학사.

1999년 한국과학기술원 전자공학과 석사.

1999년~2002년 한국전자통신연구원 반도체 원천기술연구소 연구원

2003년~현재 한국전자통신연구원 이동통신연구소 연구원.

<주관심분야: 저전력 회로 설계 기법 및 통신시스템, 이동통신용 단말 SoC 구조설계, WCDMA, SDR(Software Defined Radio), 4세대 광대역 이동통신시스템, Radio Resource Management >



배 영 환(정회원)

1985년 한양대학교 전자공학과 졸업(학사)

1987년 한양대학교 대학원 전자공학과 석사

1987년~현재 전자통신연구원 기반기술연구소 시스템IC 설계팀 책임연구원

<주관심분야: ASIP 합성, 인터페이스 합성, Em-bedded Processor 설계, SoC system 수준 설계>

한 진 호(정회원)

1998년 한국과학기술원 전기 및 전자공학과 학사 졸업

2001년 한국과학기술원 전자전산학과 전기 및 전자공학 석사 졸업

2001년~현재 한국전자통신연구원 기반기술연구소 고성능SoC연구부 시스템IC설계팀 근무

<주관심분야: SoC Platform, Configurable Pro-cessor Design, Embedded Processor Design, Low Power Circuit>

조 한 진(정회원)

1982년 한양대학교 전자공학과 졸업

1987년 5월 New Jersey Institute of Technology 전기공학과 졸업 (공학석사)

1992년 5월 University of Florida 전기공학과 졸업(공학박사)

1992년 11월~현재 한국전자통신연구원 근무. 현재 시스템IC설계팀 팀장.

<주관심분야: SoC 설계 방법론, 무선통신, 멀티미디어 설계 >